VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

TOPIC: TYPING TUTOR

DATA STRUCTURES AND ALGORITHM

(CSE2003)

(Slot: B2+TB2)

Abhishek Mukherjee (19BCE0598)

Shagun Mishra (19BCE0565)

Dr Sendhil Kumar K S

Associate Professor

Winter Semester 2019-2020

Project Description

Typing Tutor is specially designed to improve the user's typing skills in speed and accuracy. As college students, we spend a lot of time working on computers and most of the time we are in a hurry as time is of the essence. Having good typing skills helps in a big way. This led our group into making this Typing Tutor program for benefitting the students into improving their typing skills.This system works on the concept of storing and retrieving the user's typing data whenever required. It's easy to operate and to be understood by users. The user interface is very simple and user will have no trouble to navigate.

Introduction

These days, typing is a very important skill for it' s use in our day to day lives. Typing is still one of the most important computer skills you can learn. Learning to type fast and accurately helps us in many ways in life, and it should be considered an essential skill for anyone working on a computer.

Few of the reasons why typing fast is beneficial

1. Saves Time
2. Makes More Productive
3. Improves Your Posture

By this Project, we are facilitating the effective use of programming in improving typing skills.

Data Structures Concepts used

The whole project is designed in 'C++' language and different variables and data structures (Linked List using structure 'LL') , have been used for the development of this project.

The data structure Linked List has been implemented using the structure 'LL'

The linked list data structure has the following uses in the project:

1. To create a user-defined data element at the first instance.
2. To store the various attempts of user in linked lists.
3. Each level of attempt has been assigned a linked list for storing data.
4. To retrieve and display user's previous attempts data.

Methodology of Project

Framework of Project:

C++ Libraries used:

<Iostream>

<Time.h>

<Windows.h>

<Stdib.h>

<Fstream.h>

<String>

Algorithm:

- Start
- Displays this menu:
    - New user
    - Old user
    - Read records
    - Exit
- User enters username
- User selects an option from the main menu
- If user selects 1.new user, new record created for the username
- If user selects 2.old user, the user's records are only accessed for the username
- So after user types name and presses enter

- New menu opens
    - Beginner
    - Intermediate
    - Expert
    - Go Back

- User types 1 and presses enter
- Control then opens Beginner.txt, copies 1 line, and displays it
- User writes that line and presses enter
- Control then calculates time, accuracy and stores it in the linked list, then goes back to this menu
- When user presses 4, control runs through the LL, finds the smallest time for a particular group, say beginner, and stores it in Previous Games file
- Control then goes back to main menu. User maybe use any other option.
- If user presses 3, control goes to the submenu for user to select which level of difficulty he wants the records.
- Control then opens the required file, parses through it, places the data in a linked list and displays them.
- User maybe then press 4 in the Main menu, to which it ends the program.

Modules used and their description:

In FileCreator.cpp:

Create() : To create files for the 1st time.

```
void Create()

    {
       std::fstream file;
       file.open("BegPrevGames.txt", std::ios::app);
       file.close();
       file.open("IntPrevGames.txt", std::ios::app);
       file.close();
       file.open("AdvPrevGames.txt", std::ios::app);
       file.close();
    }
```

In LLops.cpp:

Insert() : To insert a node into linked list.

```
LL* Insert(LL* head, string N, double A, double T, int M)

    {
       LL * node = new LL;
       node->name = N;
       node->acc = A;
       node->time = T;
       node->menu = M;
       node->next = NULL;
       if(head == NULL) node->next = NULL;
       else
       {
          node->next = head;
       }
       head = node;
       return head;
    }
```

Delete() : Deletes a node from the list based on the name.

```
LL * Delete(LL* head, string N)

    {
       LL * node = new LL;
       node = Search(head, N);
       if(node == NULL)
       {
          cout<<"Record not found";
          return head;
       }
       LL * temp1 = node->next;
       temp1->next = node->next;
       delete temp1;
       return head;
    }
```

search() : To search a record in linked list

```
LL * Search(LL* head, string N)

    {
       LL* p = NULL;
       p = head;
       while(p != NULL)
       {
          if((p->next)->name == N)
             return p;
          p = p->next;
       }
```

```
    return NULL;
}
```

In Driver.cpp:

whereItHappens() : Method to calculate the time taken and accuracy of 1 instance of typing session

```
LL whereItHappens(string line)

{
    int c = 0;
    LL node;
    cout<<"Start writing\n\n";
    cout<<line<<endl;
    clock_t start = clock();
    string input;
    cin.ignore();
    getline(cin,input);
    clock_t en = clock();
    for(string::iterator i = line.begin(), j = input.begin(); j != input.end(); i++, j++)
    {
        if(*i != *j)
            c++;
    }
    node.acc = 1.0 - ((double)c)/line.length();
    node.time = ((double)(en - start))/CLOCKS_PER_SEC;
    cout<<"Time : "<<node.time<<endl;
    cout<<"Accuracy : "<<node.acc<<endl;
    Sleep(5000);
    return node;
}
```

SearchLeast(): Searches for the training instance of each level of difficulty which took the least time and returns it

```
LL * SearchLeast(LL * head, int menu)

{
    LL * p = head;
    LL * minnode = NULL;
    double minimum = 100000.0;
    while(p != NULL)
    {
        if(p->menu == menu)
        {
            if(p->time < minimum)
            {
                minimum = p->time;
                minnode = p;
            }
```

```
            }
            p = p->next;
        }

    return minnode;
    }
```

mainProgram() : coordinating the submenu, directing the computer to present the appropriate level of difficulty to the user and saving the result in files.

```
void mainProgram(LL* head, string Name)

    {
        int ch;
        ifstream b,i,e;
        b.open("Beginner.txt");
        i.open("Intermediate.txt");
        e.open("Expert.txt");
        do
        {
            LL node;
            system("CLS");
            ch = SubMenu();
            switch(ch)
            {
            case 1:
                {
                    string line;
                    getline(b,line);
                    cout<< "----------Tutorial----------\n\n"<< endl;
                    cout<< "Put Index finger of left hand on F and Index finger of right hand on J..."<<
    endl;
                    cout<< "The buttons have a raised bar on the bottom part\n"<< endl;
                    cout<< "Put the remaining fingers of left hand towards the left of F and that of right
    hand towards right of J"<< endl;
                    cout<< "The spacebar can be held by either of the thumbs... Use only one of
    them\n"<< endl;
                    cout<< "----------Tutorial----------\n"<< endl;
                    Sleep(10000);
                    node = whereItHappens(line);
                    break;
                }
            case 2:
                {
                    string line;
                    getline(i,line);
                    cout<< "----------Tutorial----------\n\n"<< endl;
                    cout<< "Initially, the fingers are the same as that for beginner "<< endl;
                    cout<< "The other keys can be reached by moving the fingers diagonally up or
    down\n"<< endl;
                    cout<< "----------Tutorial----------"<< endl;
                    Sleep(10000);
                    node = whereItHappens(line);
                    break;
```

```
            }
        case 3:
            {
                string line;
                getline(e,line);
                cout<< "-----------Tutorial-----------\n\n"<< endl;
                cout<< "Shift and Ctrl can be reached by stretching the left little finger"<< endl;
                cout<< "The keys to the right can be reached by stretching the right little finger\n"<<
endl;
                cout<< "The numbers can be similarly reached by stretching the other fingers"<< endl;
                cout<< "-----------Tutorial-----------"<< endl;
                Sleep(10000);
                node = whereItHappens(line);
                break;
            }
        case 4: break;
        }
        if(ch != 4) head = Insert(head,Name,node.acc,node.time,ch);
    }while(ch != 4);
    b.close();
    i.close();
    e.close();
    for(int j = 1; j <= 3; j++)
    {
        LL * temp = SearchLeast(head, j);
        ofstream y;
        if(j == 1) y.open("BegPrevGames.txt", ios:: app);
        else if(j == 2) y.open("IntPrevGames.txt", ios:: app);
        else if(j == 3) y.open("AdvPrevGames.txt", ios:: app);
        if(temp)
        {
            y<< temp->name<< "#"
             << temp->acc<< " "
             << temp->time<< "\n" ;
        }
        y.close();
    }
}
```

newUser(): Directs the computer on how to give the user a new environment for their

typing session

```
void newUser()

    {
        LL * head;
        head = NULL;
        string name;
        cout<<"Enter Username: ";
        cin.ignore();
        getline(cin,name);
        mainProgram(head,name);
    }
```

oldUser(): Directs the computer on how to give the old user a previous environment for their typing session

```
void oldUser()

    {
        Create();
        LL * head;
        head = NULL;
        string name, NameFromFile = "";
        cout<<"Enter Username: ";
        cin.ignore();
        getline(cin, name);
        ifstream f;
        int fl = 0;
        bool fbeg, fint, fexp; fbeg = fint = fexp = false;
        f.open("BegPrevGames.txt");
        while(true)
        {
            getline(f,NameFromFile,'#');
            //cout<< NameFromFile;
            if(NameFromFile == name)
            {
                cout<< "You have a save in Beginner"<<endl;
                fl = 1;
                fbeg = true;
                break;
            }
            if(f.eof()) break;
            getline(f,NameFromFile);
        }
        f.close();
        f.open("IntPrevGames.txt");
        while(true)
        {
            getline(f,NameFromFile,'#');
            //cout<< NameFromFile;
            if(NameFromFile == name)
            {
                cout<< "You have a save in Intermediate"<<endl;
                fl = 1;
                fint = true;
                break;
            }
            if(f.eof()) break;
            getline(f,NameFromFile);
        }
        f.close();
        f.open("AdvPrevGames.txt");
        while(true)
```

```
    {
        getline(f,NameFromFile,'#');
        //cout<< NameFromFile;
        if(NameFromFile == name)
        {
            cout<< "You have a save in Expert"<<endl;
            fl = 1;
            fexp = true;
            break;
        }
        if(f.eof()) break;
        getline(f,NameFromFile);
    }
    f.close();
    int ch = 0;
    if(fl == 0)
    {
        cout<< "You don't have a save progress...\nPress 1 to create a new user file or 0 to go back:
";
        cin>> ch;
        if(ch == 1)
        {
            newUser();
            return;
        }
        else if(ch == 0) return;
        else
        {
            cout<< "Invalid Input. Please Enter Correct Input";
            goto point1;
        }
    }
    double AccFromFile; double TimeFromFile;
    cout<< "Select the category to open"<<endl<<endl;
    point1:
    ch = SubMenu();
    switch(ch)
    {
        case 1:
            {
                if(fbeg)
                    f.open("BegPrevGames.txt");
                else
                {
                    cout<< "You dont have a save in Beginner"<<endl;
                    goto point1;
                }
                break;
            }
        case 2:
            {
                if(fint)
                    f.open("IntPrevGames.txt");
                else
                {
                    cout<< "You dont have a save in Intemediate"<<endl;
```

```
                goto point1;
            }
            break;
        }
    case 3:
        {
            if(fexp)
                f.open("AdvPrevGames.txt");
            else
            {
                cout<< "You dont have a save in Expert"<<endl;
                goto point1;
            }
            break;
        }
    case 4: return;
    default:
        {
            cout<< "Invalid Input. Please Enter Correct Input"<<endl;
            goto point1;
        }
    }
    getline(f,NameFromFile,'#');
    f>> AccFromFile;
    f>> TimeFromFile;
    head = Insert(head, NameFromFile, AccFromFile, TimeFromFile, ch);
    mainProgram(head, name);
}
```

viewRecords(): Instructs the computer to display the previous records of all the users that have used the tutorial

```
void viewRecords()

{
    LL * head, * Min; int i = 1;
    head = NULL;
    string NameFromFile; double AccFromFile; double TimeFromFile;
    ifstream f;
    point2:
    system("CLS");
    int ch = SubMenu();
    switch(ch)
    {
        case 1: {f.open("BegPrevGames.txt"); break; }
        case 2: {f.open("IntPrevGames.txt"); break; }
        case 3: {f.open("AdvPrevGames.txt"); break; }
        default: {cout<< "Invalid Input. Please Enter Correct Input."; goto point2; }
    }
    while(true)
    {
        getline(f,NameFromFile,'#');
        if(f.eof()) break;
        f>> AccFromFile;
```

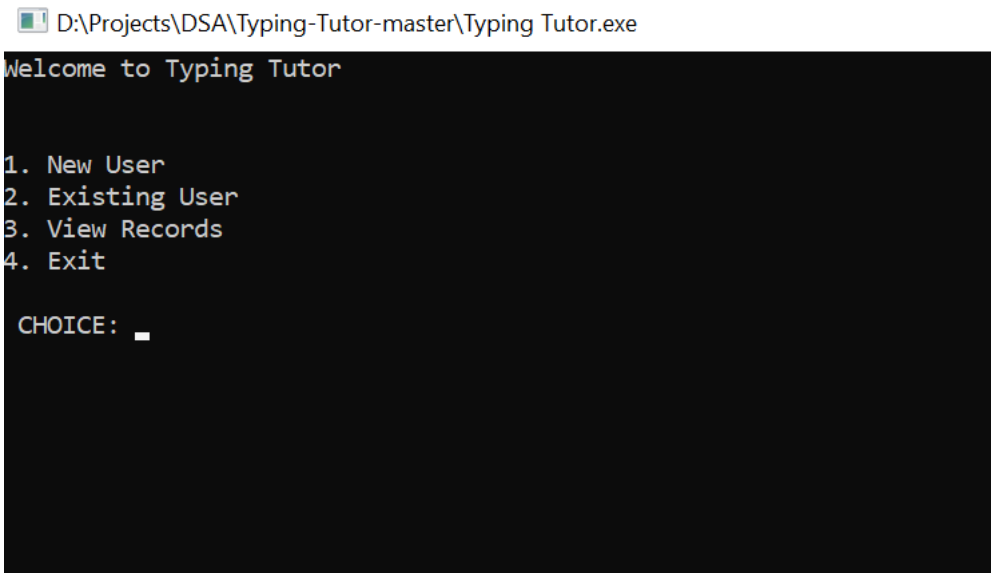
```
            f>> TimeFromFile;
            head = Insert(head, NameFromFile, AccFromFile, TimeFromFile, ch);
        }
        cout<< "Sl. No.\t\tName\t\tTime\t\tAccuracy\n";
        Min = head;
        while(Min)
        {
            cout<< i << "\t\t" << Min->name << "\t\t" << Min->time << "\t\t" << Min->acc << endl;
            Min = Min->next;
            i++;
        }
        cout<< "Press any key to return: ";
        string anyKey;
        cin>> anyKey;
    }
```

main() : Main driver method that is executed at the beginning. It displays the main menu and coordinates all other methods.

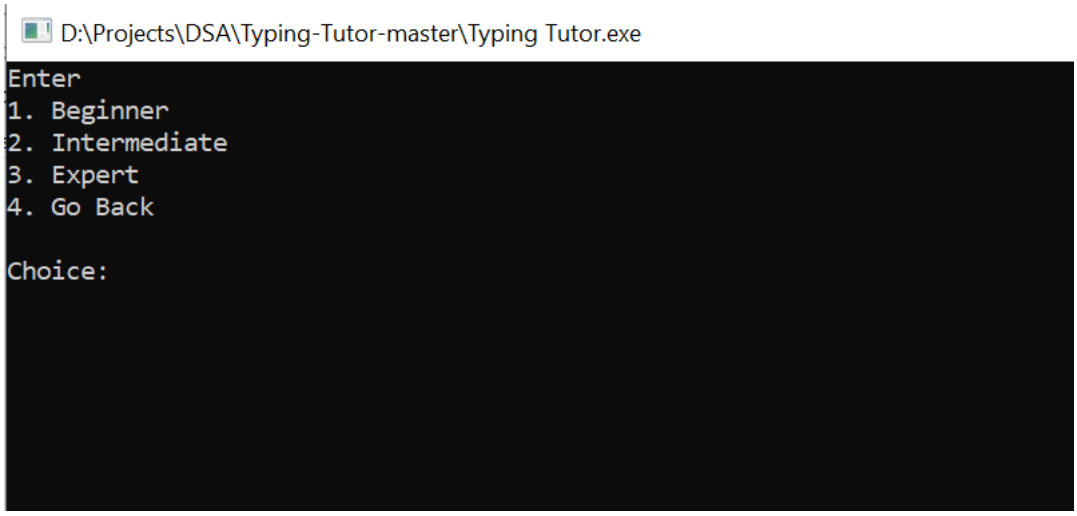
```
int main()

    {
        while(true)
        {
            system("CLS");
            switch(Menu())
            {
            case 1:
                newUser();
                break;
            case 2:
                oldUser();
                break;
            case 3:
                viewRecords();
                break;
            case 4:
                exit(0);
            default: cout<<"Input Entered is invalid, please enter a Valid Input"<<endl;
            }
        }
    }
```

Complete Demonstration of Project

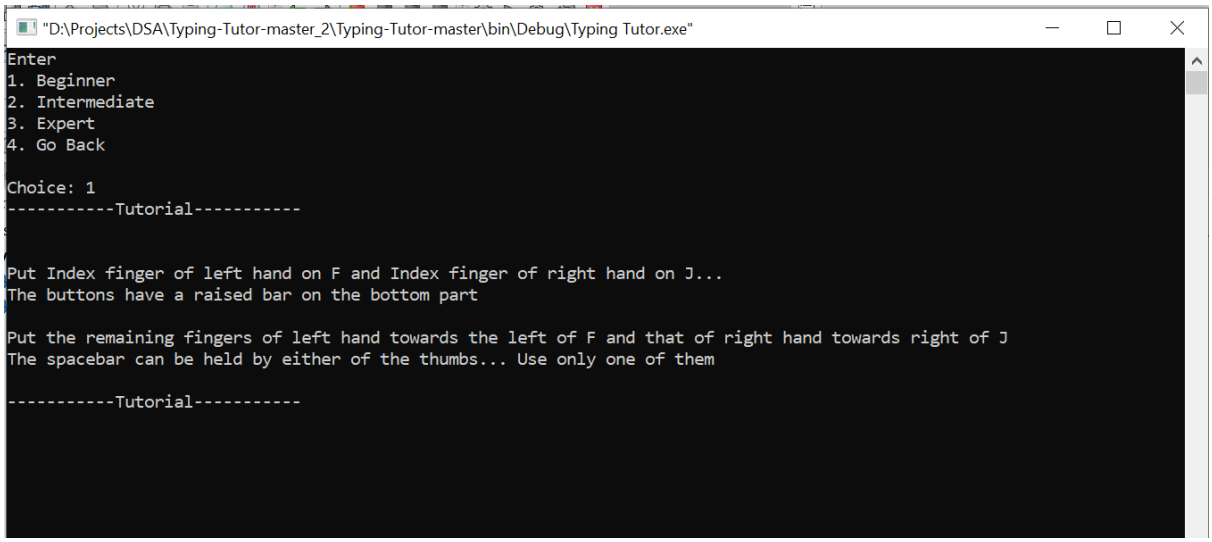
Main Menu:



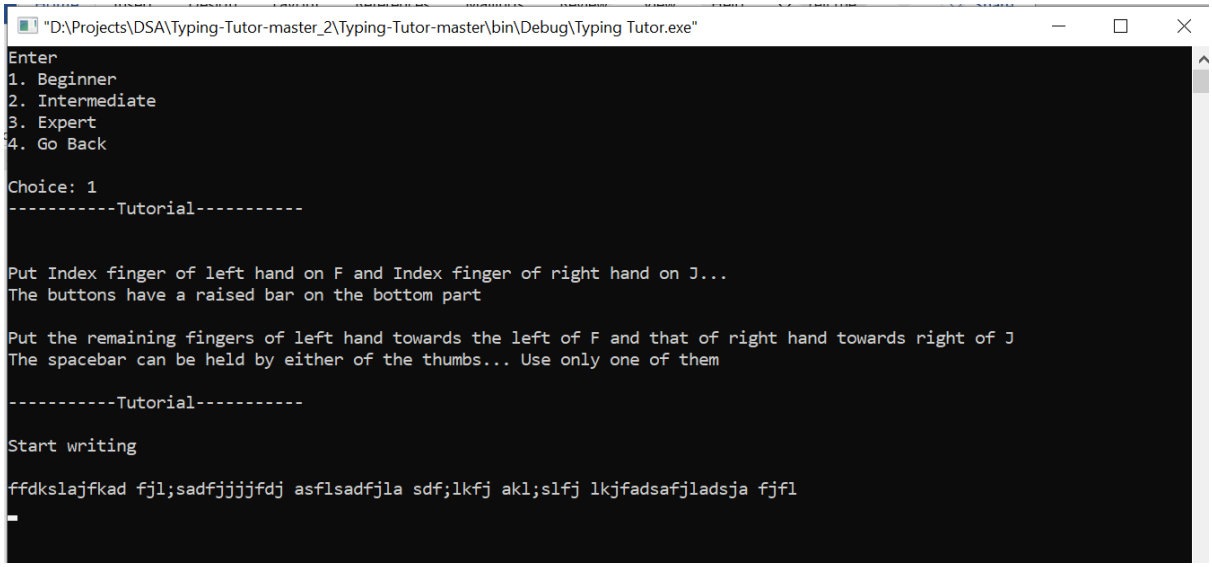
Sub Menu:



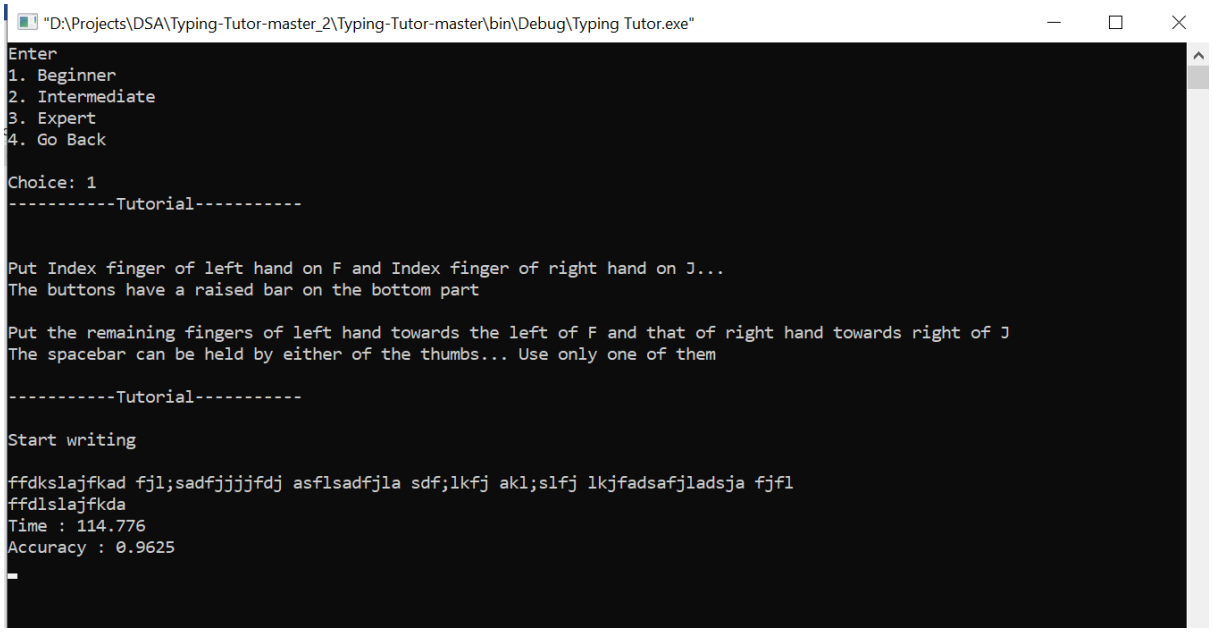
Beginner Level Attempt (Tutorial):



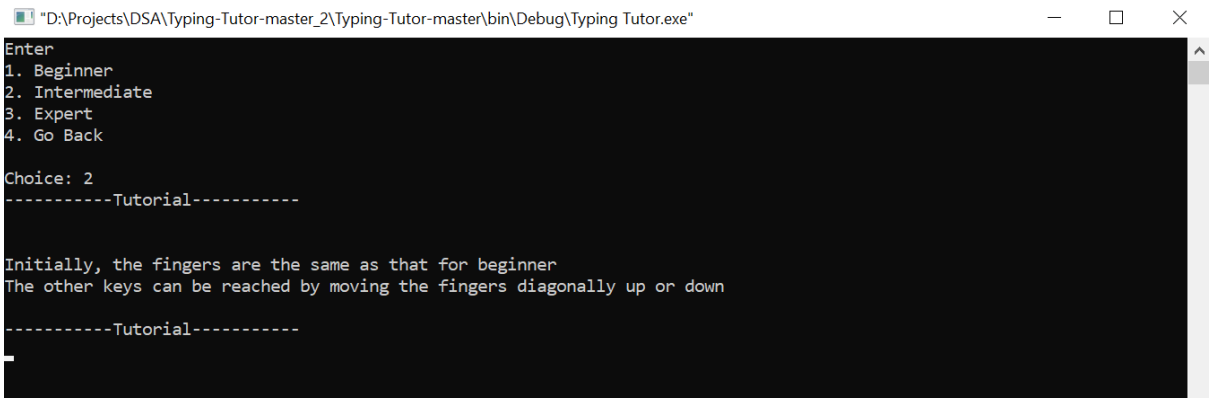
Beginner Level attempt:



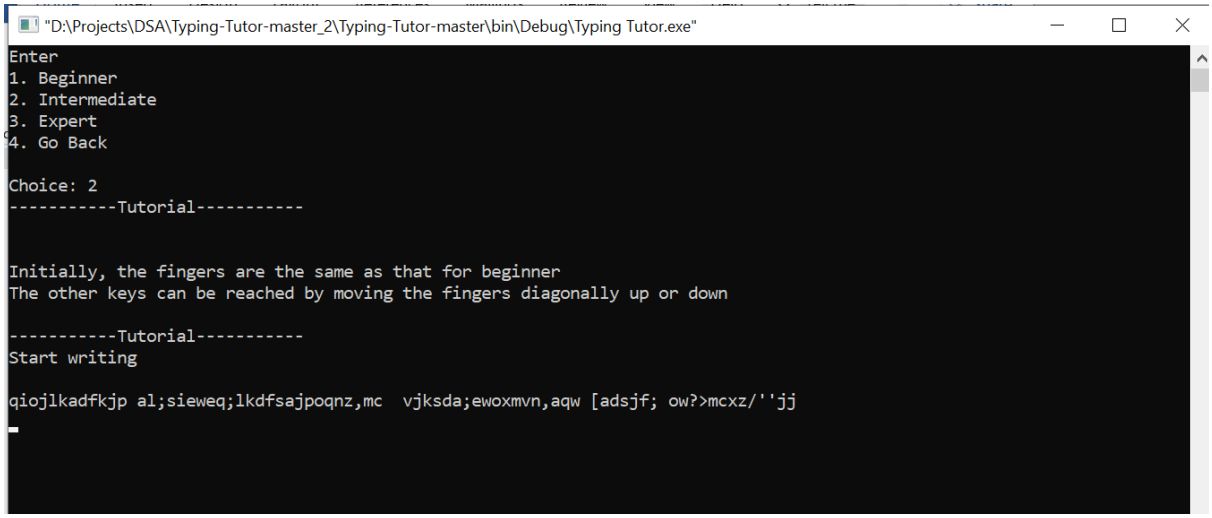
Displaying beginner attempt results:



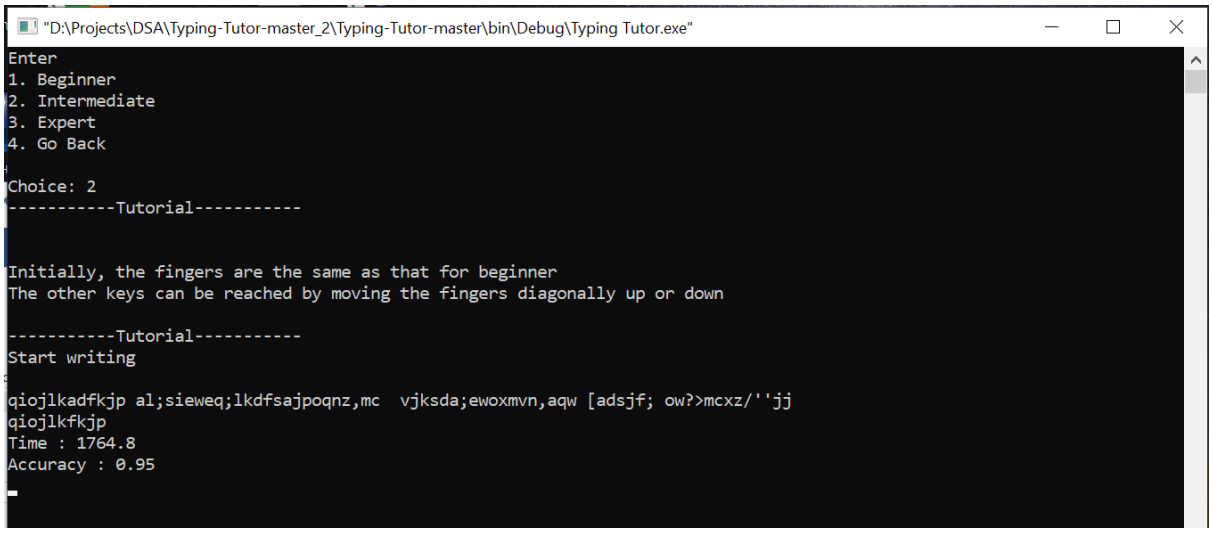
Intermediate Level attempt (Tutorial) :



Intermediate Level attempt:

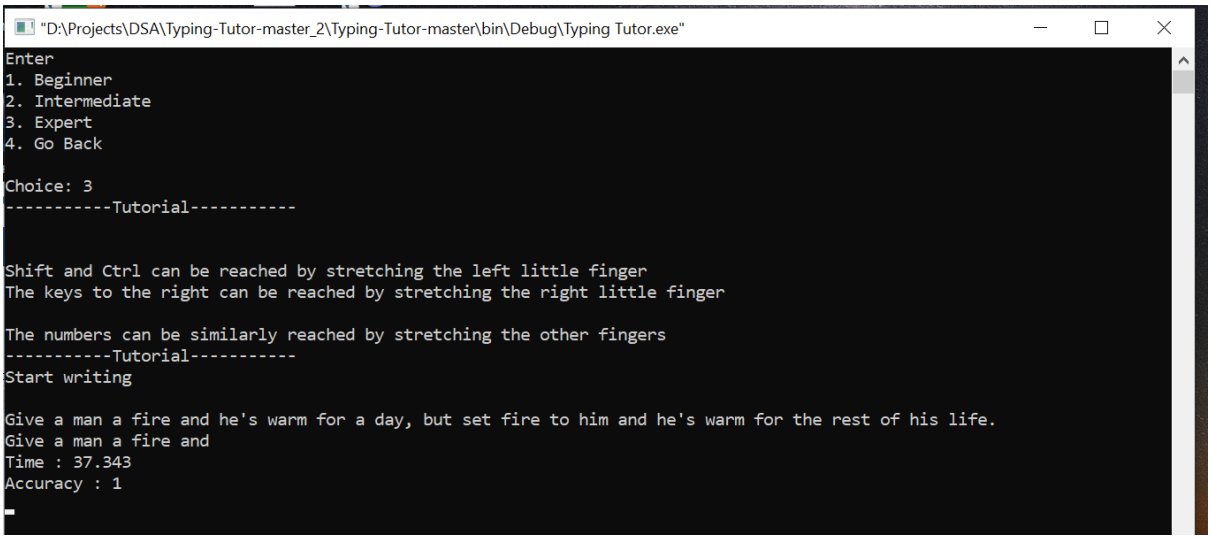


Displaying Intermediate attempt results:



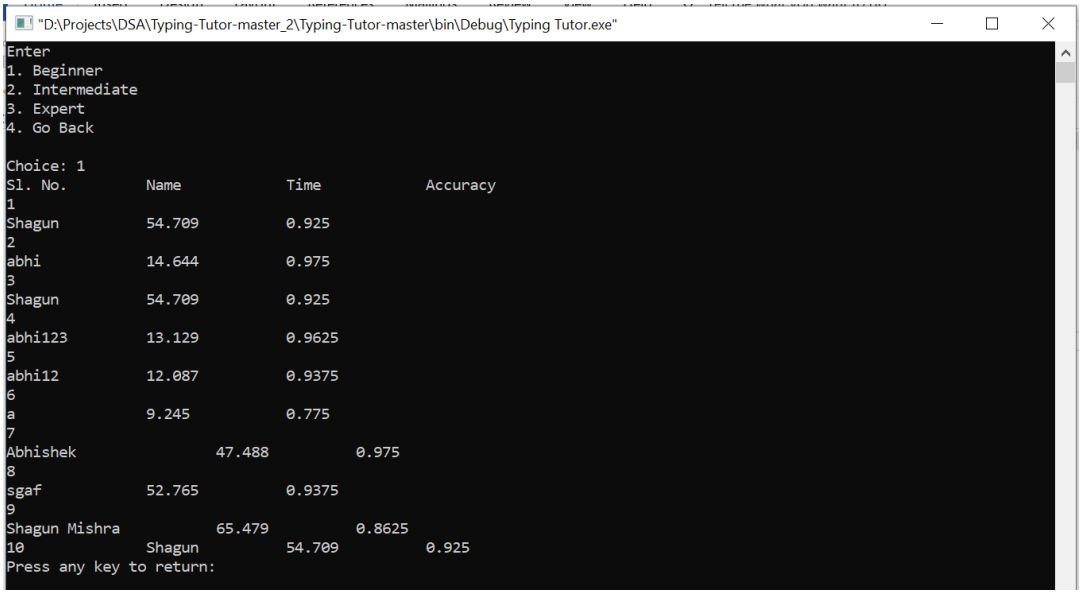
Expert Level attempt (Tutorial) :

Expert Level attempt with results:

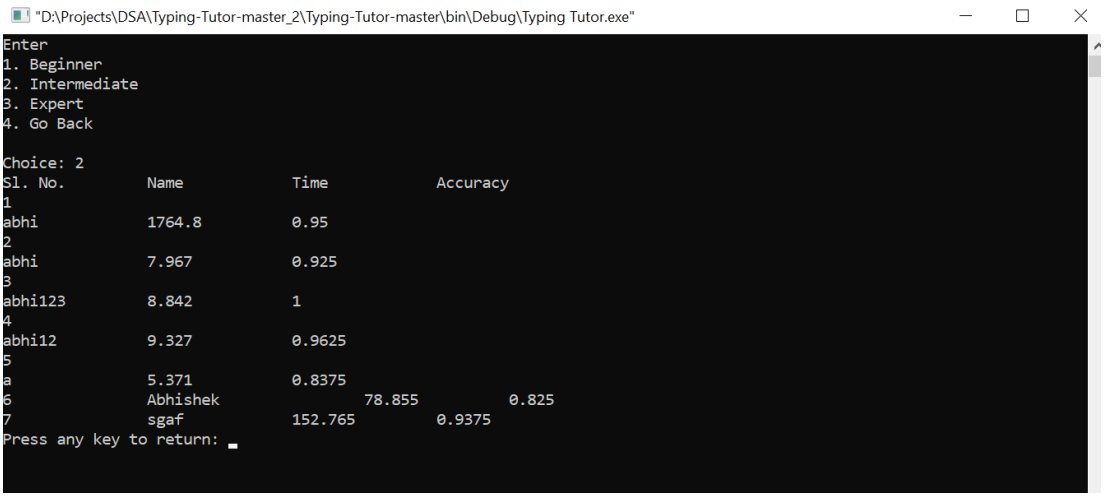


<u>Viewing Records:</u>

Beginner level:



Intermediate Level:



Conclusion:

Such a project will be very beneficial for every student in improving their typing skills.

This is a fundamental skills for everybody to help in their day to day lives.

It'll help in developing the productivity and time management skills significantly.

References:

Weblinks:

1. https://www.geeksforgeeks.org/data-structures/linked-list/
2. https://www.geeksforgeeks.org/timer-c-using-system-calls/
3. https://www.w3schools.com/cpp/cpp_files.asp
4. https://adamfortgo.wordpress.com/2013/11/18/5-tips-that-help-to-pass-a-wpm-typing-test/

Books:

1. Introduction to Algorithms, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, 3rd edition
2. Data Structures and Algorithms made easy, Narasimha Karumanchi, 5th edition